

(12)

(19)

(11)

2 236 880₍₁₃₎A

(43) Date of A publication 17.04.1991

(22) Date of filing 13.09.1990

(32) 02.10.1989

(33) IE

(Incorporated in Ireland)

(72) Inventors
Neil Wilson
Steven John Metzler

(74) Agent and/or Address for Service
Lloyd Wise Tregear & Co
Norman House, 105-109 Strand, London, WC2R 0AE,
United Kingdom

(52) UK CL (Edition K)
G4A AFI

US 3829839 A

(58) Field of search
UK CL (Edition K) G4A AFI
INT CL⁶ G06F 9/46
Online database: WPI

(57) A method for controlling the operation of a computer to handle interrupts comprises comparing the priority of an interrupt with the priority of a task being carried out by the computer. The task being carried out is suspended if the priority of the interrupt is higher than that of the task, and the interrupt is run. An interrupt counter is incremented by 1 on each interrupt being run. On completion of an interrupt, the counter is decremented by one and the computer is returned to the task which had been interrupted. A reschedule flag is set on an interrupt calling for a rescheduling of the tasks being carried out in the foreground process of the computer. A reschedule flag causes the foreground process to be rescheduled before the computer returns to the foreground process.



GB 2 236 880 A

Foreground process

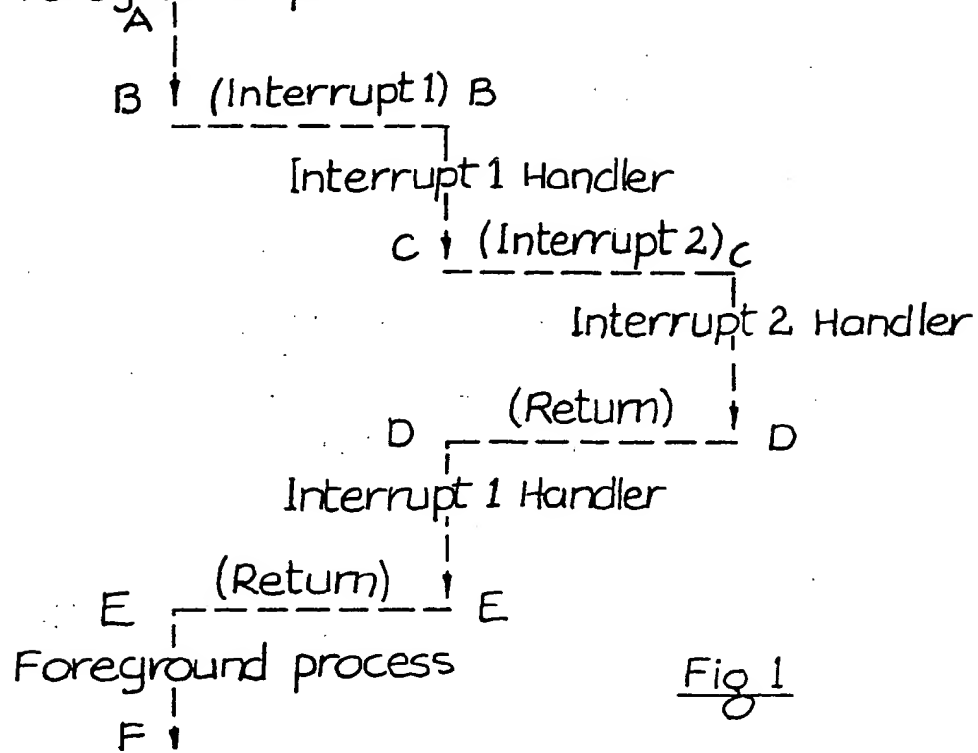


Fig 1

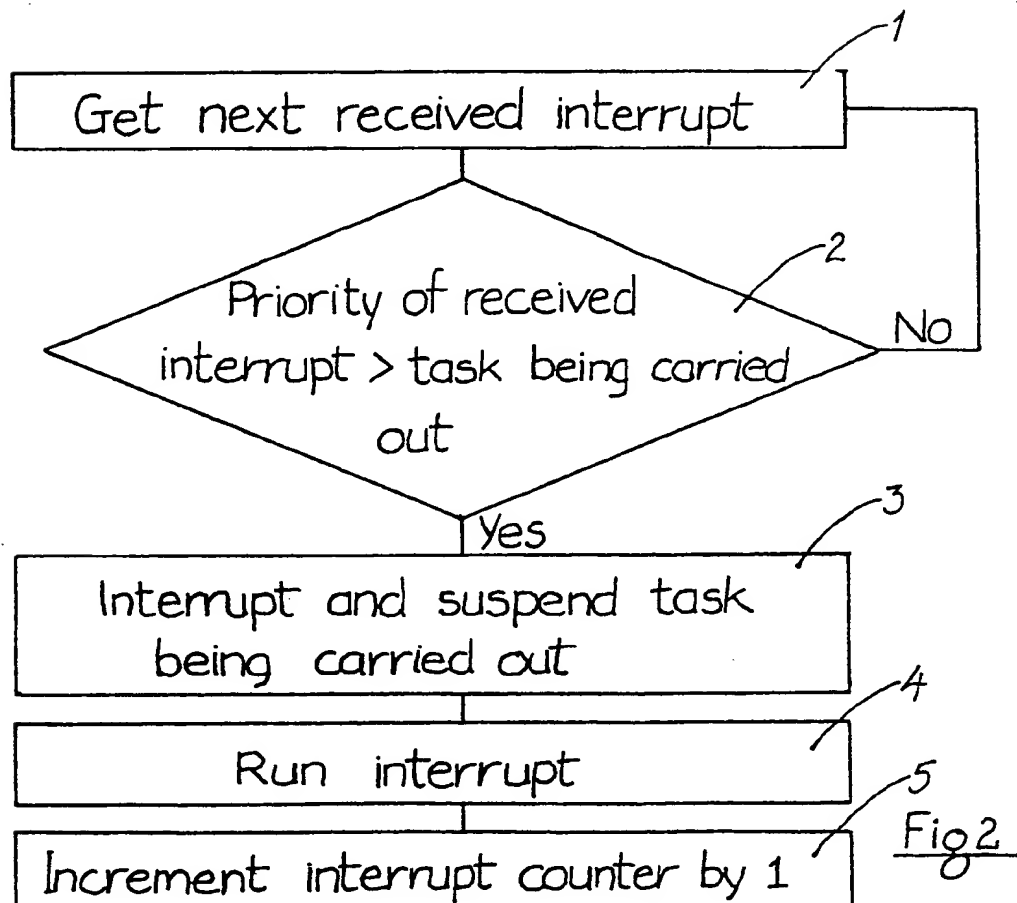
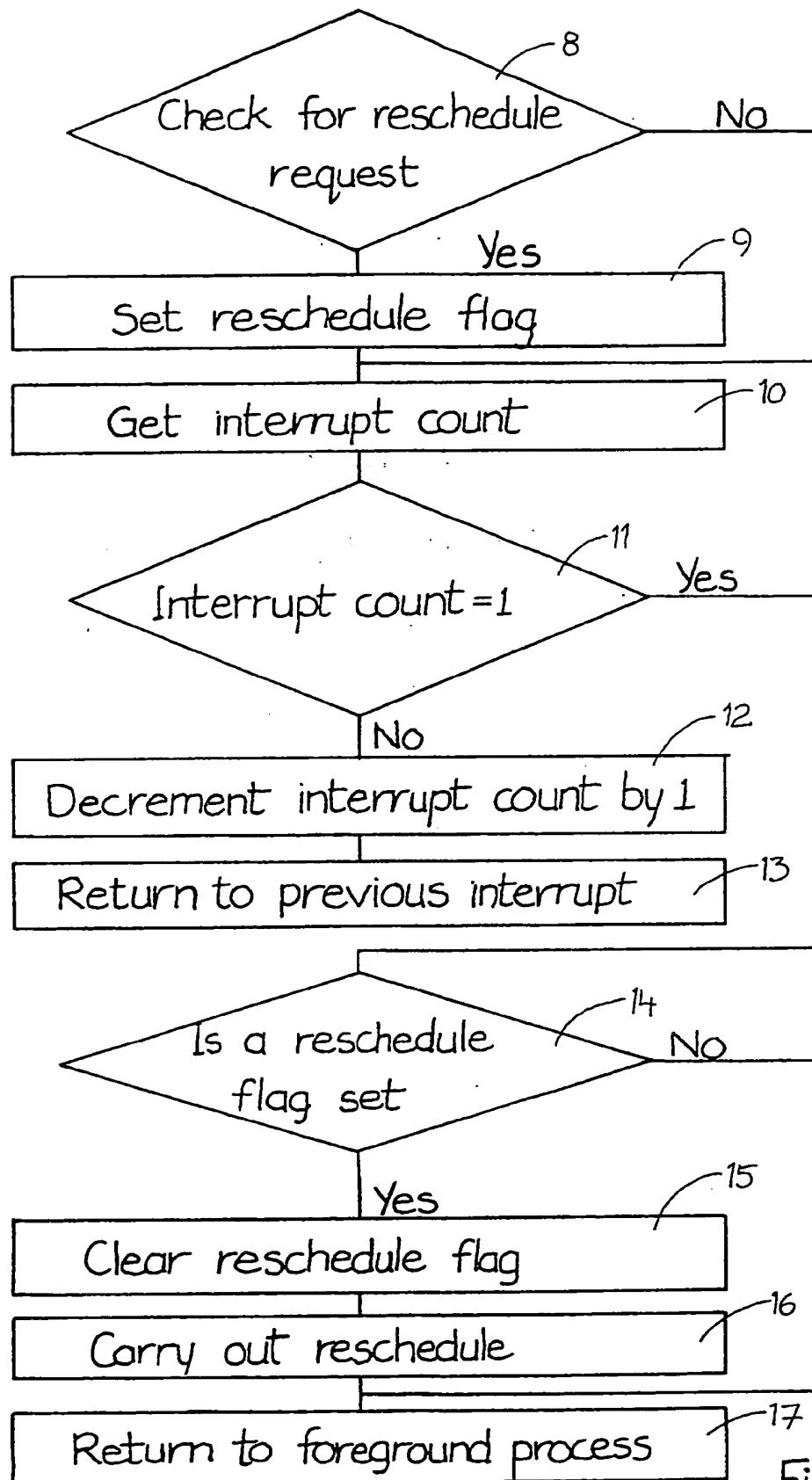
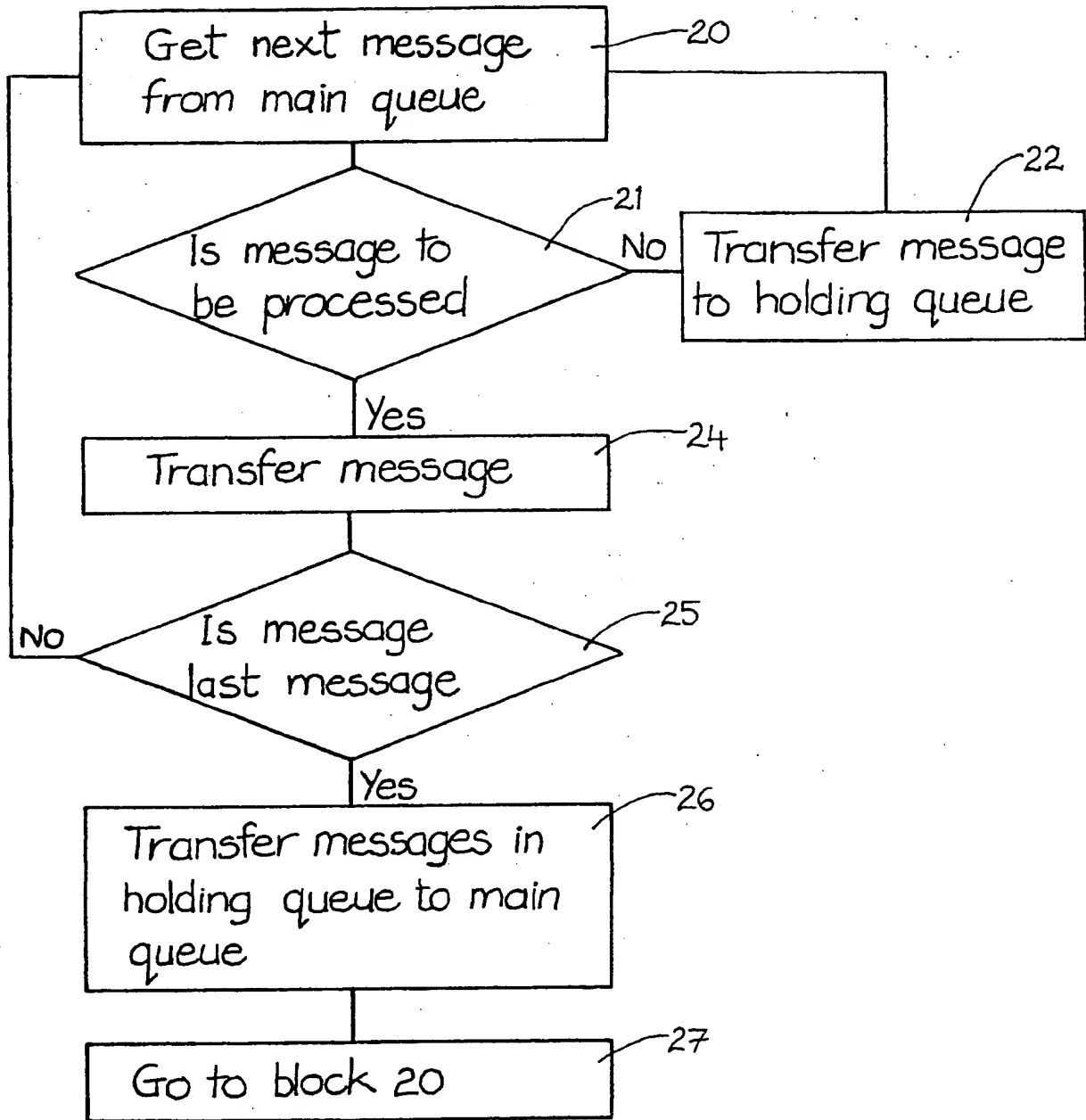


Fig 2

Fig 3

Fig 4

4/4

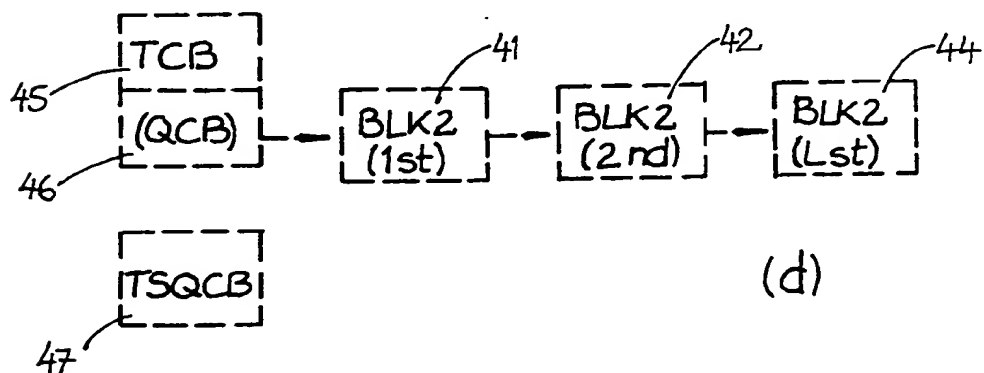
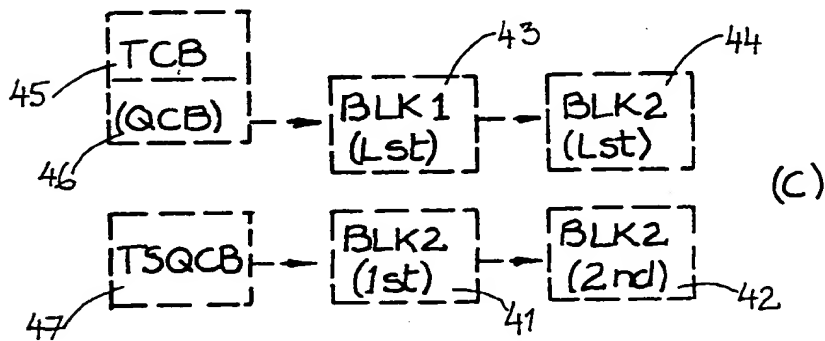
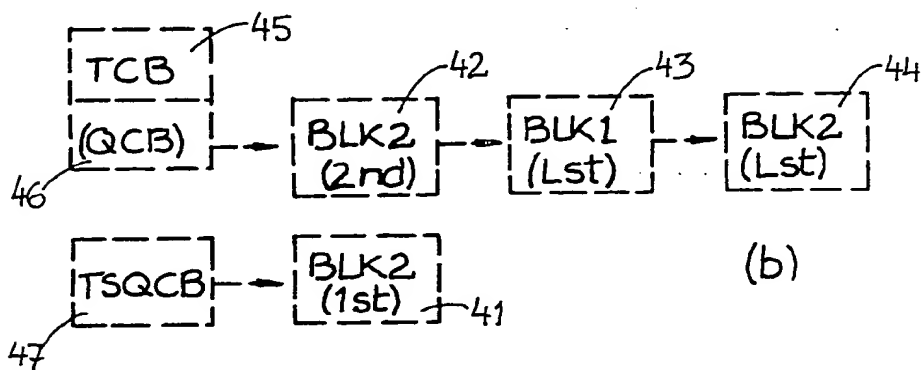
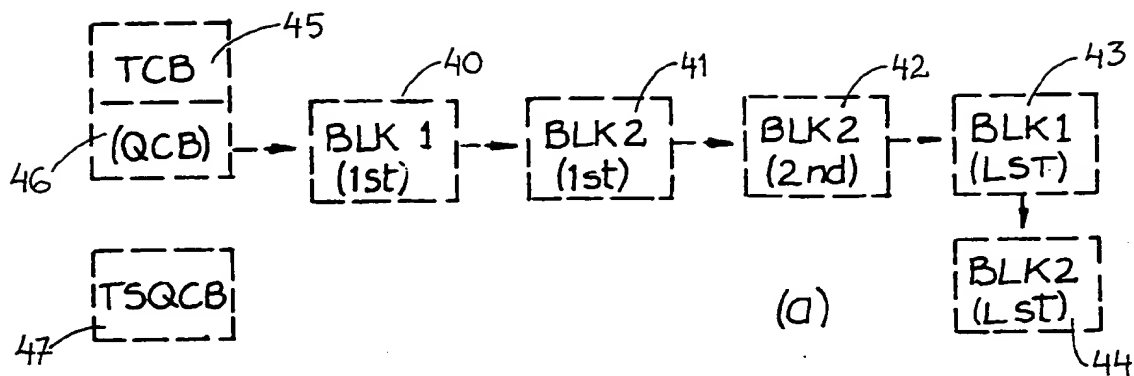


Fig 5

2236880

- 1 -

**Title: A METHOD FOR CONTROLLING THE OPERATION
OF A COMPUTER TO HANDLE INTERRUPTS**

The present invention relates to a method for
controlling the operation of a computer to handle
interrupts. The invention also relates to a method for
controlling the operation of a computer for enabling the
5 messages of a block of messages to be selected from a
queue of messages of a plurality of blocks. Further,
the invention relates to a computer and to a computer
programme for controlling the computer.

According to the invention, there is provided a method
10 for controlling the operation of a computer to handle a
plurality of interrupts of increasing priority while a
foreground process is suspended, the method comprising
the steps of enabling an interrupt with a higher
priority than the task being executed to interrupt the
15 task being executed, running the interrupt with the
higher priority, incrementing an interrupt counter by
one each time an interrupt commences to run, on
completion of each interrupt decrementing the counter by
one, setting a reschedule flag to indicate that an
20 interrupt has called for rescheduling of the tasks to be
carried out by the computer, on completion of all
interrupts searching for a reschedule flag, rescheduling
the tasks in the computer in response to a rescheduling
flag being found, returning the computer to a foreground
25 process with the highest priority which is ready to run
on all interrupts having been completed. Preferably,

the method includes the step of calling up a subroutine to increment the interrupt counter on commencement of an interrupt running.

- In one embodiment of the invention, the method further
- 5 searches for one or more messages of a block of messages for processing from a main queue which comprises a plurality of blocks of messages, the messages of the said block being distributed through the main queue, the method further comprising the steps of sequentially
- 10 selecting the messages in the main queue, establishing if the selected message is of the block to be processed, and if the message is the last message of the said block, transferring the message to the destination to be processed if the message is a message of the said block,
- 15 transferring each message not of the said block, found prior to finding the last message of the said block into a holding queue, and maintaining the order of the messages in the holding queue similar to their order in the main queue.
- 20 Preferably, the messages in the holding queue are transferred to the beginning of the main queue on the last message of the said block having been transferred for processing.

Additionally, the invention provides a computer having

means to receive a plurality of interrupts, and means for assigning a priority for execution to each interrupt, the computer further comprising means for interrupting a task being executed to permit an
5 interrupt with a higher priority to run, means for incrementing an interrupt counter by one each time an interrupt commences to run, means for decrementing the interrupt counter by one on completion of each interrupt, means for setting a reschedule flag to
10 indicate that an interrupt has called for the scheduling of the tasks to be carried out by the computer, searching means for searching for a reschedule flag on completion of all interrupts, rescheduling means for rescheduling the tasks in the computer, the rescheduling
15 means being responsive to a rescheduling flag being found, means for returning the computer to a foreground process with the highest priority which is ready to run.

In a further embodiment of the invention, the computer further comprises a plurality of buffers for storing a
20 plurality of respective messages or blocks of messages, the messages being stored in the form of a main queue, the computer further comprising means for searching for one or more messages of a block to be processed, the search means comprising means for sequentially selecting
25 the messages from the buffers in the order of the main queue, means for establishing if a selected message is a

message of the said block and if the message is the last message of the said block, means for transferring each message of the said block to a destination to be processed, and means for transferring each message not
5 of the said block found prior to the last message of the said block to a holding queue, and means for maintaining the order of the messages in the holding queue similar to their order in the main queue.

In another embodiment of the invention, means for
10 transferring the messages from the holding queue to the main queue is provided, the means for transferring the messages from the holding queue to the main queue being responsive to the last message of the said block being transferred.

15 Additionally, the invention provides a computer programme which comprises the method of the invention.

Additionally, the invention provides a medium comprising the computer programme of the invention.

The invention will be more clearly understood from the
20 following description of a preferred embodiment thereof, given by way of example only, with reference to the accompanying drawings, in which:

Fig. 1 is a schematic view of the operation of a computer under the control of a computer programme of the invention,

5 Fig. 2 is a flow chart of a subroutine of the computer programme according to the invention,

Fig. 3 is a flow chart of another subroutine of the computer programme according to the invention,

Fig. 4 is a flow chart of another subroutine of the computer programme according to the invention, and

10 Figs. 5(a) to (d) are schematic representations of the operation of the computer under the computer programme of the invention.

Referring to the drawings, there is illustrated computer programme subroutines for controlling the operation of a
15 computer to handle interrupts and for selecting messages from a main queue of messages. The subroutines for dealing with interrupts operate so that on an interrupt being received by the computer and being assigned a priority rating by the computer, the received interrupt
20 can interrupt the task currently being undertaken by the computer if the interrupt has a higher priority rating than the task being carried out by the computer which

may be another interrupt or a foreground process. This is particularly advantageous when the computer is being used in a real time communications environment, which demands fast response to external events. The computer programme subroutines for selecting messages from the main queue of messages enables one or more messages of a block of messages to be selected for processing from the queue of messages, and permits the messages to be transferred for processing without disturbing the order of the messages. The subroutines are called up by a main computer programme which controls the operation of the computer which is not illustrated, however, such computers and computer programmes will be known to those skilled in the art. Only the subroutines of the computer programme which form the invention are described in detail. For clarity, the two sets of subroutines will be dealt with separately.

In general, it is envisaged that the computer programme subroutines for carrying out the method of the invention will be provided in a computer. However, they may be provided on a suitable medium, such as for example a magnetic medium, for example on a floppy disc, magnetic tape, micro-disc or the like.

Dealing initially with the interrupt subroutine and referring to Figs. 1 to 3, Fig. 1 illustrates

graphically the operation of the subroutine, flow charts of which are illustrated in Figs. 2 and 3. The computer programme comprises a subroutine for assigning a priority to each interrupt received. This subroutine is not described, however such subroutines will be known to those skilled in the art. On an interrupt being assigned a priority rating, the priority rating of the interrupt is compared with the priority rating of the foreground process running or task running on the computer. If the interrupt is of a higher priority rating, then the foreground process or task running on the computer is suspended.

Referring in particular to Fig. 1, in this case the task being carried out in the foreground process has a priority rating 0 and is running for the period AB. An interrupt, namely interrupt 1 with priority rating 1 is received. The interrupt 1 is of higher priority rating than the foreground process task, which causes the foreground process task to be suspended at time B. The interrupt 1 of priority rating 1 commences to run at time B and continues to time C, when a second interrupt, namely, interrupt 2 of priority rating 2, in other words, a priority rating higher than the first interrupt is received. Interrupt 1 being the task running on the computer is interrupted at time C and interrupt 2 commences to run from time C and runs until it is

completed to time D. On completion, the computer
reverts to the previous task running, which in this case
is interrupt 1, which runs for the time period DE at
which stage it is completed. The computer then reverts
5 at time E to a task in the foreground process which is
described below.

Referring to Fig. 2, the subroutine for interrupting a
task running on the computer is illustrated. Block 1
gets the received interrupt. Block 2 compares the
10 priority rating of the interrupt with the priority
rating of the task running on the computer. If the
priority rating of the interrupt is less than that of
the task running on the computer, the subroutine returns
to block 1. If the priority rating of the interrupt is
15 greater than the priority rating of the task running on
the computer, the subroutine goes to block 3 which makes
the interrupt and suspends the task currently running on
the computer. The subroutine then moves onto block 4,
which runs the interrupt. Block 5 increments an
20 interrupt counter by one, thereby keeping a record of
the number of interrupts running and partly completed.
The interrupt continues to run until it is completed or
until it is interrupted by an interrupt of a higher
priority. In this case, where an interrupt, namely
25 interrupt 2 of a higher priority than interrupt 1 is
received, the interrupt 1 is interrupted as already

described.

On each interrupt being completed, the subroutine of Fig. 3 is called up. Block 8 checks if the interrupt is calling for the task being executed in the foreground process of the computer to be rescheduled. For example, if an interrupt is the last of a number of messages dealing with a particular task, and on completion of the interrupt if the task is ready to be run in the foreground process, the tasks in the foreground process should be rescheduled to run the task of which the interrupt forms part in the appropriate sequence with the tasks of the foreground process. If rescheduling is requested, the subroutine moves to block 9 which sets a rescheduling flag. The programme then moves to block 10 which gets the value of the current interrupt count. If a reschedule is not called for, the subroutine moves directly to block 10. Block 11 checks if the interrupt count is 1. In other words, have all the interrupts been completed. If the interrupt count is not equal to 1, the subroutine moves to block 12, which decrements the interrupt count by 1, and the subroutine then moves to block 13, which returns the computer to execute the previously interrupted interrupt.

If block 11 determines that the interrupt count is 1, in other words, if the last interrupt has just been

executed, the computer programme moves to block 14 which searches for a rescheduling flag. If no rescheduling flag is found, the computer programmes moves to block 17, which returns the computer to the foreground process. On the other hand, if a rescheduling flag is found, the subroutine moves to block 15 which clears the rescheduling flag and in turn to block 16, which carries out the reschedule requested. After the reschedule has been completed, the subroutine moves to block 17. Thus, the computer is returned to the foreground process, which operates in accordance with the reschedule. If the task of which any of the interrupts calling for a reschedule forms part has a higher priority than the suspended task in the foreground process, then that task with the higher priority is run before the suspended task of the foreground process. On the other hand, if that task has a lower priority than the suspended task of the foreground process, then the foreground process continues with the suspended task and the task of which the interrupt forms part is processed in sequence in accordance with its priority.

Referring now to Figs. 4 and 5 the message selection subroutine will now be described. A flow chart of the subroutine is illustrated in Fig. 4 while the movement of messages within the computer is illustrated in Figs. 5(a) to (d). In this case, data and information is

stored in blocks in the computer. Each block comprises a plurality of messages indicated by the reference numerals 40 to 44. The messages 40 to 44 are stored in buffers in the computer. The messages 40 to 44 could typically be received by the computer from a communications network. The messages as they are received are stored in buffers and queued in a main queue, see Fig. 5(a). The computer operating under the main computer programme executes each operation as a task. A task has an associated task control block 45 which is used to maintain the status of the task, see Fig. 5. Each task control block 45 also contains an embedded queue control block 46. The queue control block 46 holds messages in the main queue which are received from other tasks. A task specific queue control block 47 of each task holds messages in a holding queue described below. Where the task running on the computer requires all the messages of a block, say block 1, the subroutine of Fig. 4 is called up. Block 20 gets the first message in the main queue, namely message 40. Block 21 establishes if the message 40 is of the block to be processed. If the first message is not of block 1, the subroutine moves to block 22, which stores the message in a holding queue formed by temporary storage buffers under the control of the task specific queue control block 47, see Fig. 5(b). The messages stored in the holding queue are stored in

the same order as they had in the main queue. The subroutine then returns to block 20 to get the next message in the main queue. Where the comparison of block 21 shows that the message 40 is of block 1, the
5 subroutine moves to block 24. Block 24 transfers the message to a destination in the computer where the task is running for processing. The subroutine then moves to block 25. Block 25 checks if the message being received is the last message in the block. If it is not the last
10 message, the subroutine returns to block 20 to get the next message sequentially in the queue. If the message is the last message, the subroutine moves to block 26, which transfers the messages stored in the holding queue back to the beginning of the main queue in the sequence
15 in which they originally were. The subroutine then moves to block 27, which returns the subroutine to block 20.

Referring now specifically to Figs. 5(a) to (d), the operation of the subroutine of Fig. 4 will be described.
20 Initially, the subroutine selects the first message, namely message 40 from the main queue. Since this is the first message of block 1, which is the required block, the first message 40 of block 1 is transferred to the appropriate task for processing. The subroutine
25 then selects the next message 41 from the main queue, which in this case is the first message of block 2,

which is transferred to the first position of the holding queue, see Fig. 5(b). The subroutine then selects the next message 42 from the main queue, which is the second message of block 2, and this is likewise
5 moved to the holding queue, and in this case is assigned to the second position in the holding queue, see Fig. 5(c). The subroutine then selects the next message 43 from the main queue, which in this case is the last message for block 1. This is transferred to the
10 appropriate task. Since all the messages of block 1 have now been selected and transferred, the subroutine under the control of block 26 transfers all the messages in the holding queue, namely the first and second messages 41 and 42 of block 2 back into the main queue.
15 It can be seen from Figs. 5(a) to (d) that in all cases the messages 40 to 44 are transferred from the main queue to the holding queue so that when in the holding queue, they are in the same order in which they were in the main queue. Similarly, when the messages 40 to 44
20 are transferred back from the holding queue to the main queue, the remaining messages in the main queue take up the same order they had prior to the messages being transferred from the main queue to the holding queue. In other words, the messages on being transferred from
25 the holding queue to the main queue take up the same order in the main queue as they had in the holding queue.

The subroutine of Fig. 4 then commences to select the first message in the main queue, which is now the first message 41 of block 2. Thus, the subroutine according to the invention permits messages of different blocks to
5 be moved from the main queue to the holding queue, while at the same time maintaining the messages in their chronological order, and permitting them to be returned to the main queue in that chronological order after the messages of a particular block have been dealt with.

10 While the computer has been described as operating under the control of the computer programme having a subroutine to handle interrupts and to handle queuing, it will be appreciated that in all cases it is not necessary that the queue handling subroutine need be
15 provided.

CLAIMS

1. A method for controlling the operation of a computer to handle a plurality of interrupts of increasing priority while a foreground process is suspended, the
5 method comprising the steps of enabling an interrupt with a higher priority than the task being executed to interrupt the task being executed, running the interrupt with the higher priority, incrementing an interrupt counter by one each time an interrupt commences to run,
10 on completion of each interrupt decrementing the counter by one, setting a reschedule flag to indicate that an interrupt has called for rescheduling of the tasks to be carried out by the computer, on completion of all interrupts searching for a reschedule flag, rescheduling
15 the tasks in the computer in response to a rescheduling flag being found, returning the computer to a foreground process with the highest priority which is ready to run, on all interrupts having been completed.

2. A method as claimed in Claim 1 in which the method
20 includes the step of checking if an interrupt is calling for a rescheduling of the tasks to be carried out by the computer.

3. A method as claimed in Claim 1 or 2 in which the tasks to be rescheduled are tasks being carried out in
25 the foreground process.

4. A method as claimed in any preceding claim in which on rescheduling of the tasks being carried out the reschedule flag is cleared.

5. A method as claimed in any preceding claim in which
5 the method further searches for one or more messages of a block of messages for processing from a main queue which comprises a plurality of blocks of messages, the messages of the said block being distributed through the main queue, the method further comprising the steps of
10 sequentially selecting the messages in the main queue, establishing if the selected message is of the block to be processed, and if the message is the last message of the said block, transferring the message to the destination to be processed if the message is a message
15 of the said block, transferring each message not of the said block, found prior to finding the last message of the said block into a holding queue, and maintaining the order of the messages in the holding queue similar to their order in the main queue.

20 6. A method as claimed in Claim 5 in which the messages in the holding queue are transferred to the beginning of the main queue on the last message of the said block having been transferred for processing.

7. A method as claimed in Claim 6 in which the order of the messages in the holding queue is maintained on the messages being transferred to the main queue.

8. A method for controlling the operation of a
5 computer, the method being substantially as described herein with reference to and as illustrated in the accompanying drawings.

9. A computer having means to receive a plurality of interrupts, and means for assigning a priority for
10 execution to each interrupt, the computer further comprising means for interrupting a task being executed to permit an interrupt with a higher priority to run, means for incrementing an interrupt counter by one each time an interrupt commences to run, means for
15 decrementing the interrupt counter by one on completion of each interrupt, means for setting a reschedule flag to indicate that an interrupt has called for the scheduling of the tasks to be carried out by the computer, searching means for searching for a reschedule
20 flag on completion of all interrupts, rescheduling means for rescheduling the tasks in the computer, the rescheduling means being responsive to a rescheduling flag being found, means for returning the computer to a foreground process with the highest priority which is
25 ready to run.

10. A computer as claimed in Claim 9 in which means for comparing the priority of an interrupt received with an interrupt running is provided.

11. A computer as claimed in any of Claims 9 or 10 in
5 which the computer is operated under the control of a computer programme which carries out the method of any of Claims 1 to 4.

12. A computer as claimed in any of Claims 9 to 11 in
10 which the computer further comprises a plurality of buffers for storing a plurality of respective messages of blocks of messages, the messages being stored in the form of a main queue, the computer further comprising means for searching for one or more messages of a block to be processed, the search means comprising means for
15 sequentially selecting the messages from the buffers in the order of the main queue, means for establishing if a selected message is a message of the said block and if the message is the last message of the said block, means for transferring each message of the said block to a
20 destination to be processed, and means for transferring each message not of the said block found prior to the last message of the said block to a holding queue, and means for maintaining the order of the messages in the holding queue similar to their order in the main queue.

13. A computer as claimed in Claim 12 in which means
for transferring the messages from the holding queue to
the main queue is provided, the means for transferring
the messages from the holding queue to the main queue
5 being responsive to the last message of the said block
being transferred.

14. A computer as claimed in Claim 13 in which the
messages in the holding queue are transferred to the
main queue in the order in which they are stored in the
10 holding queue.

15. A computer substantially as described herein with
reference to and as illustrated in the accompanying
drawings.

16. A computer programme which comprises the method
15 steps as claimed in any of Claims 1 to 8.

17. A computer programme suitable for running in a
computer of any of Claims 9 to 15.

18. A computer programme substantially as described
herein with reference to and as illustrated in the
20 accompanying drawings.

19. A medium comprising the computer programme of any of Claims 16 to 18.

20. A medium as claimed in Claim 19 in which the medium is a magnetic medium.

5 21. A medium as claimed in Claim 19 in which the medium is a floppy disc or tape.